

# **FOOD DELIVERY TIME PREDICTOR**

*Dissertation submitted in fulfilment of the requirements for the Degree of*

## **BACHELOR OF TECHNOLOGY**

### **in COMPUTER SCIENCE AND ENGINEERING**

By

**SHAHANA T**

**12219504**

Supervisor

**VED PRAKASH CHAUBEY**



### **School of Computer Science and Engineering**

Lovely Professional University

Phagwara, Punjab (India)

Month ..... Year .....

@ Copyright LOVELY PROFESSIONAL UNIVERSITY, Punjab (INDIA)

Month ..... Year .....

## DECLARATION STATEMENT

---

I hereby declare that the research work reported in the dissertation/dissertation proposal entitled "FOOD DELIVERY TIME PREDICTOR" in partial fulfilment of the requirement for the award of Degree for Bachelor of Technology in Computer Science and Engineering at Lovely Professional University, Phagwara, Punjab is an authentic work carried out under supervision of my research supervisor Mr. Ved Prakash Chaubey. I have not submitted this work elsewhere for any degree or diploma.

I understand that the work presented herewith is in direct compliance with Lovely Professional University's Policy on plagiarism, intellectual property rights, and highest standards of moral and ethical conduct. Therefore, to the best of my knowledge, the content of this dissertation represents authentic and honest research effort conducted, in its entirety, by me. I am fully responsible for the contents of my dissertation work.

*Signature of Candidate*

**Shahana**

**R. No 12219504**

## TABLE OF CONTENTS

### CONTENTS

### PAGE NO.

---

Abstract	4
Introduction	4
Problem Statement	5
Solution Approach	6
Result Analysis ~ Code ~ Code Output	8-21
Conclusion	21
Reference	22

## **ABSTRACT**

This article explores the practical application of Machine Learning algorithms to achieve precise food delivery time predictions within the context of renowned food delivery services like Zomato and Swiggy. The primary aim is to establish a strong foundation for transparency with customers by providing them with accurate delivery time estimates. In the following sections, we offer a comprehensive guide to developing a robust food delivery time prediction system using Machine Learning techniques in conjunction with the Python programming language.

## **INTRODUCTION**

Food delivery services have evolved to become a cornerstone of modern living, offering unparalleled convenience and a vast array of culinary options at one's doorstep. However, timely and reliable deliveries remain a fundamental aspect of customer satisfaction in this highly competitive industry. Leading food delivery giants, including Zomato and Swiggy, understand that accurate delivery time predictions are crucial for ensuring customer delight.

### **Objectives:**

**Precise Delivery Time Estimates:** The core objective of this article is to demonstrate how Machine Learning can be leveraged to predict food delivery times with a high degree of precision. Accurate time predictions foster transparency and customer trust.

**Empowering the Food Delivery Industry:** We aim to empower food delivery companies and data enthusiasts by providing a step-by-step guide to developing their food delivery time prediction system. This knowledge opens up new horizons for innovation in the industry.

Enhancing Customer Experience: The ultimate goal is to enhance the customer experience. Timely deliveries, backed by reliable predictions, not only meet customer expectations but also have the potential to exceed them, thereby boosting customer loyalty.

### **Scope:**

The scope of this article encompasses the following areas:

*Machine Learning Application:* We delve into the application of Machine Learning algorithms to address the food delivery time prediction challenge. The techniques outlined here can be adopted by food delivery companies and data practitioners to develop their predictive systems.

*Python Programming:* All methodologies presented are implemented using the Python programming language, making it accessible to a broad audience. Python's versatility and extensive libraries are instrumental in creating robust Machine Learning models.

*Data Science Best Practices:* Our approach adheres to data science best practices, encompassing data collection, preprocessing, feature engineering, model selection, and deployment. Readers can learn and apply these principles in their own projects.

*Transparency and Customer Satisfaction:* The concept of transparency is central to our discussion. We emphasize the importance of setting accurate expectations for customers and ensuring a seamless and delightful food delivery experience.

As we delve into the intricacies of food delivery time prediction using Machine Learning, we invite readers to embark on a journey that promises to redefine how the food delivery industry caters to the diverse needs of its customers. In the following sections, we will provide practical insights, methodologies, and future directions for those eager to unlock the potential of data-driven delivery predictions.

## **PROBLEM STATEMENT**

The central challenge in food delivery time prediction is to calculate the estimated time of arrival for a given order. This entails determining the

distance between the restaurant and the delivery location and identifying the historical correlation between delivery time and distance travelled.

### 1. Distance Calculation:

One of the fundamental aspects is to determine the distance between the food preparation point (usually a restaurant or kitchen) and the point of food consumption (the customer's delivery location). Accurately quantifying this distance is pivotal to providing precise delivery time estimates.

### 2. Historical Time-Distance Relationship:

An indispensable element is to establish a robust historical correlation between the delivery time and the distance travelled. This historical data is derived from past delivery orders with varying distances. Analysing this data enables the prediction system to understand how delivery times tend to vary concerning different distances.

### 3. Real-Time Factors:

The food delivery industry operates in real-time, and several dynamic factors can impact delivery times, such as traffic conditions, weather, and the availability of delivery personnel. Integrating real-time data into the prediction model is an essential aspect of addressing the problem.

### 4. Customer Expectations:

Beyond technical considerations, understanding and managing customer expectations play a pivotal role. Customers expect reliable and transparent delivery time estimates. The system must align with these expectations to maintain customer trust and satisfaction.

## **SOLUTION APPROACH**

### **Data Collection:**

The food delivery time prediction process commences with the collection of historical data from past deliveries. This dataset includes a plethora of

parameters, such as the geographical coordinates of the restaurant's location, the customer's delivery address, the actual time taken for delivery, and various other relevant factors. This historical dataset serves as the bedrock for developing our food delivery time prediction model.

### **Data Preprocessing:**

The collected dataset undergoes an extensive data preprocessing phase to render it suitable for Machine Learning analysis. This crucial stage involves several key tasks:

- **Data Cleaning:** The dataset is cleansed of any inconsistencies, inaccuracies, or outliers that could adversely affect the model's performance.
- **Handling Missing Values:** Any missing or null values are addressed to ensure that the dataset is complete and reliable.
- **Categorical Data Conversion:** Categorical data, such as city names, is transformed into a format that Machine Learning models can interpret. This typically involves techniques like one-hot encoding.

### **Feature Engineering:**

In addition to considering the straight-line distance between the restaurant and the delivery location, our model incorporates additional features that can significantly impact delivery times. These features may include:

- **Time of Day:** Recognizing that traffic conditions and delivery times vary throughout the day.
- **Weather Conditions:** Weather-related factors, such as rain or snow, can influence delivery efficiency.
- **Traffic Density:** The model takes into account the real-time traffic density for accurate predictions.

By introducing these additional features into the model, we enhance its ability to make precise delivery time estimates.

### **Model Selection:**

Selecting the appropriate Machine Learning model is a pivotal step in solving the prediction task. Regression models are commonly utilized for time prediction problems, and in this article, we opt for the XGBoost regression model. The XGBoost model is known for its robustness and predictive accuracy, making it a suitable choice for our food delivery time prediction system.

### **Model Training:**

The chosen model is then trained on the historical delivery data. The dataset is divided into training and testing sets, enabling us to assess the model's performance. During this phase, we use evaluation metrics such as mean squared error (MSE), mean absolute error (MAE), and others to gauge the model's predictive accuracy. Techniques such as hyperparameter tuning and cross-validation are employed to optimize the model's performance and ensure it can generalize well to new, unseen data.

### **Model Deployment:**

Upon successful training and evaluation, the Machine Learning model is ready for deployment in a real-time prediction system. When a customer places an order, this system takes into account the customer's location and the restaurant's coordinates to calculate the distance. Additionally, it considers various features, including the time of day, weather conditions, and traffic density. Using these inputs, the system generates an accurate estimate of the expected delivery time and conveys it to the customer. This real-time prediction mechanism ensures that customers receive precise and trustworthy delivery time estimates.

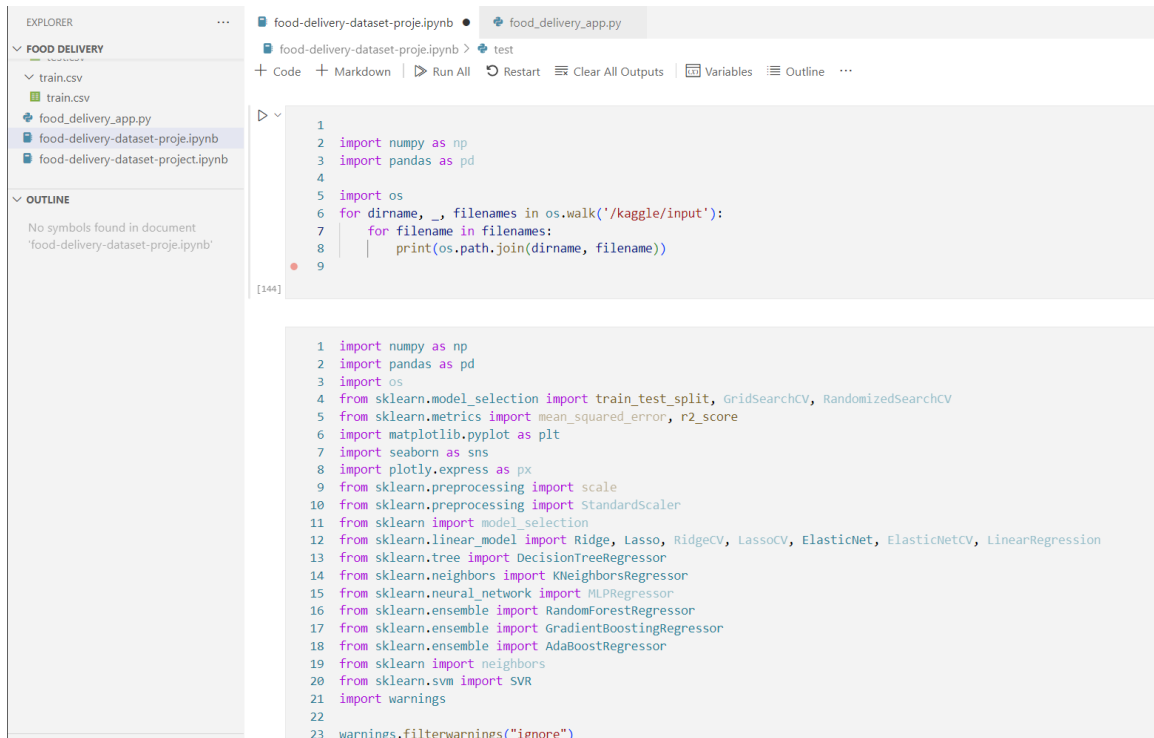
## **RESULT ANALYSIS**

Post-deployment, we rigorously analyse the performance of our prediction system. Real-world predictions are compared against actual delivery times to assess the system's accuracy. User feedback is collected and analysed to further refine the model.

The system's efficiency can be measured using various metrics, and improvements are made based on these results. Over time, the system becomes increasingly accurate in predicting food delivery times.

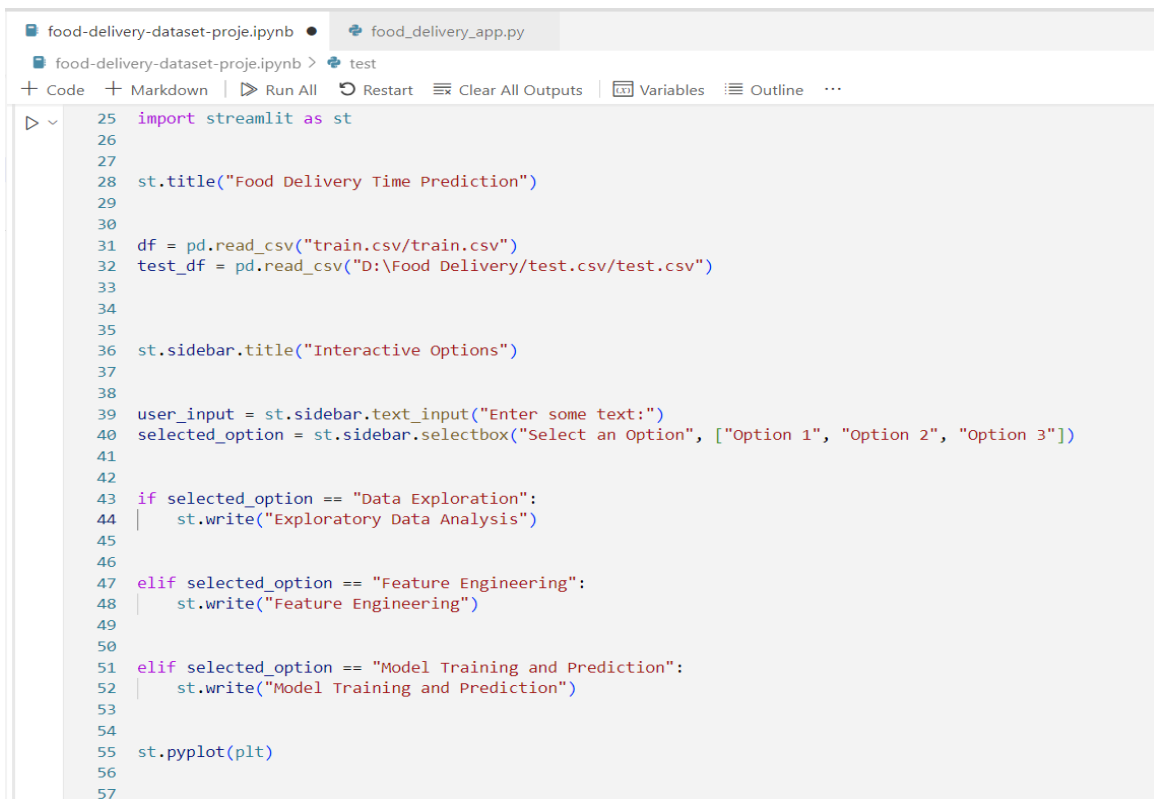


## Code



```
1
2 import numpy as np
3 import pandas as pd
4
5 import os
6 for dirname, __, filenames in os.walk('/kaggle/input'):
7     for filename in filenames:
8         print(os.path.join(dirname, filename))
9
```

```
1 import numpy as np
2 import pandas as pd
3 import os
4 from sklearn.model_selection import train_test_split, GridSearchCV, RandomizedSearchCV
5 from sklearn.metrics import mean_squared_error, r2_score
6 import matplotlib.pyplot as plt
7 import seaborn as sns
8 import plotly.express as px
9 from sklearn.preprocessing import scale
10 from sklearn.preprocessing import StandardScaler
11 from sklearn import model_selection
12 from sklearn.linear_model import Ridge, Lasso, RidgeCV, LassoCV, ElasticNet, ElasticNetCV, LinearRegression
13 from sklearn.tree import DecisionTreeRegressor
14 from sklearn.neighbors import KNeighborsRegressor
15 from sklearn.neural_network import MLPRegressor
16 from sklearn.ensemble import RandomForestRegressor
17 from sklearn.ensemble import GradientBoostingRegressor
18 from sklearn.ensemble import AdaBoostRegressor
19 from sklearn import neighbors
20 from sklearn.svm import SVR
21 import warnings
22
23 warnings.filterwarnings("ignore")
```



```
25 import streamlit as st
26
27
28 st.title("Food Delivery Time Prediction")
29
30
31 df = pd.read_csv("train.csv/train.csv")
32 test_df = pd.read_csv("D:\Food Delivery/test.csv/test.csv")
33
34
35
36 st.sidebar.title("Interactive Options")
37
38
39 user_input = st.sidebar.text_input("Enter some text:")
40 selected_option = st.sidebar.selectbox("Select an Option", ["Option 1", "Option 2", "Option 3"])
41
42
43 if selected_option == "Data Exploration":
44     st.write("Exploratory Data Analysis")
45
46
47 elif selected_option == "Feature Engineering":
48     st.write("Feature Engineering")
49
50
51 elif selected_option == "Model Training and Prediction":
52     st.write("Model Training and Prediction")
53
54
55 st.pyplot(plt)
56
57
```

```
food-delivery-dataset-proje.ipynb • food_delivery_app.py
food-delivery-dataset-proje.ipynb > test
+ Code + Markdown | Run All Restart Clear All Outputs Variables Outline ...

1 df=pd.read_csv("train.csv/train.csv")
2 df.head()

[ ]

1 df.tail()

[ ]

1 test_df=pd.read_csv("D:\Food Delivery/test.csv/test.csv")
2 test_df.head()

[ ]

1 test_df.tail()

[ ]

1 df.drop(["ID","Delivery_person_ID"],axis=1,inplace=True)
2 df.head()

[ ]

1 test_df.drop(["ID","Delivery_person_ID"],axis=1,inplace=True)
2 test_df.info()

[ ]

1 test_df.head()

[ ]

1 df.info()
```

```
food-delivery-dataset-proje.ipynb • food_delivery_app.py
food-delivery-dataset-proje.ipynb > test
+ Code + Markdown | Run All Restart Clear All Outputs Variables Outline ...

1 df["Time_taken(min)"] = df["Time_taken(min)"].str.replace("(min)","")
2

[33]

1 df["Weatherconditions"] = df["Weatherconditions"].str.replace("conditions","")
2

[34]

1 df.head()

[ ]

1 test_df["Weatherconditions"] = test_df["Weatherconditions"].str.replace("conditions","")
2

[36]

1 df.info()

[ ]

1 df.shape

[ ]

1 test_df.info()

[ ]

1 test_df.shape

[40]
```

```
food-delivery-dataset-proje.ipynb • food_delivery_app.py
food-delivery-dataset-proje.ipynb > col2=test_df.columns.tolist()
+ Code + Markdown | ▶ Run All ⌂ Restart ☰ Clear All Outputs | 📄 Variables ☰ Outline ...

[41] 1 col=df.columns.tolist()

▶ 1 col

[ ]

▶ 1 for i in col:
2     ...
3     ...print(df[i].value_counts())
4     ...print("***39)

[ ]

1 df["City"]=df["City"].str.replace("NaN","other")
2 df["multiple_deliveries"]=df["multiple_deliveries"].str.replace("NaN","1")
3 df["Road_traffic_density"]=df["Road_traffic_density"].str.replace("NaN","Low")
4 df["Festival"]=df["Festival"].str.replace("NaN","No")
5 df["Weatherconditions"]=df["Weatherconditions"].str.replace("NaN","other")

[44]

▶ 1 df.info();

[ ]

▶ 1 col2=test_df.columns.tolist()

[46]

▶ 1 col2

[ ]
```

```
food-delivery-dataset-proje.ipynb • food_delivery_app.py
food-delivery-dataset-proje.ipynb > col2=test_df.columns.tolist()
+ Code + Markdown | ▶ Run All ⌂ Restart ☰ Clear All Outputs | 📄 Variables ☰ Outline ...

▶ 1 for i in col2:
2     ...
3     ...print(test_df[i].value_counts())
4     ...print("***39)

[ ]

1 test_df["City"]=test_df["City"].str.replace("NaN","other")
2 test_df["multiple_deliveries"]=test_df["multiple_deliveries"].str.replace("NaN","1")
3 test_df["Road_traffic_density"]=test_df["Road_traffic_density"].str.replace("NaN","Low")
4 test_df["Festival"]=test_df["Festival"].str.replace("NaN","No")
5 test_df["Weatherconditions"]=test_df["Weatherconditions"].str.replace("NaN","other")

[49]

▶ 1 test_df.info();

[ ]

▶ 1 sns.distplot(df.Delivery_person_Age)
2     ...

[ ]

▶ 1 df.Delivery_person_Age=df.Delivery_person_Age.str.replace("NaN","32")
2 df.info()
3

[ ]

▶ 1 sns.distplot(df.Delivery_person_Ratings)

[ ]
```

```
food-delivery-dataset-proje.ipynb • food_delivery_app.py
food-delivery-dataset-proje.ipynb > col2=test_df.columns.to_list()
+ Code + Markdown | ▶ Run All ↺ Restart ≡ Clear All Outputs | 📄 Variables ≡ Outline ...

1
2 df.Delivery_person_Ratings=df.Delivery_person_Ratings.str.replace("NaN","4.8")
3 df.head()

[ ]

1 df.Time_Orderd=df.Time_Orderd.replace('NaN',float(np.nan),regex=True)
2

[55]

1 df.info()

[ ]

1 df.dropna(subset=['Time_Orderd'],inplace=True)
2 df.info()

[ ]

1 df.head()

[ ]

1
2 sns.distplot(test_df.Delivery_person_Age)
3

[ ]

1 test_df.Delivery_person_Age=test_df.Delivery_person_Age.str.replace("NaN","30")
2 df.info()
3
```

```
food-delivery-dataset-proje.ipynb • food_delivery_app.py
food-delivery-dataset-proje.ipynb > col2=test_df.columns.to_list()
+ Code + Markdown | ▶ Run All ↺ Restart ≡ Clear All Outputs | 📄 Variables ≡ Outline ...

1
2 sns.distplot(test_df.Delivery_person_Ratings)

[ ]

1
2 test_df.Delivery_person_Ratings=test_df.Delivery_person_Ratings.str.replace("NaN","4.8")
3 test_df.head()

[ ]

1 test_df.Time_Orderd=test_df.Time_Orderd.replace('NaN',float(np.nan),regex=True)
2

[63]

1 test_df.info()

[ ]

1
2 test_df.dropna(subset=['Time_Orderd'],inplace=True)
3 test_df.info()

[ ]

1 test_df.head()

[ ]

1 df["Delivery_person_Age"] = pd.to_numeric(df["Delivery_person_Age"], errors='coerce')
2 df["Delivery_person_Ratings"] = pd.to_numeric(df["Delivery_person_Ratings"], errors='coerce')
3 df["multiple_deliveries"] = pd.to_numeric(df["multiple_deliveries"], errors='coerce')
4
```

food-delivery-dataset-proje.ipynb

food\_delivery\_app.py

food-delivery-dataset-proje.ipynb

col2=test\_df.columns.to\_list()

Code

Markdown

Run All

Restart

Clear All Outputs

Variables

Outline

```

1 df['Time_taken(min)']=df['Time_taken(min)'].str.split("-", expand=True)[1]
2

```

[68]

1 df.head()

[ ]

```

1 df["Time_taken(min)"] = pd.to_numeric(df["Time_taken(min)"], errors='coerce')

```

[70]

1 df.info()

[ ]

```

1 test_df["Delivery_person_Age"] = pd.to_numeric(test_df["Delivery_person_Age"], errors='coerce')
2 test_df["Delivery_person_Ratings"] = pd.to_numeric(test_df["Delivery_person_Ratings"], errors='coerce')
3 test_df["multiple_deliveries"] = pd.to_numeric(test_df["multiple_deliveries"], errors='coerce')
4

```

[72]

1 test\_df.head()

[ ]

1

2 df.describe([0.15,0.25,0.35,0.5,0.65,0.75,0.9]).T

[ ]

food-delivery-dataset-proje.ipynb

food\_delivery\_app.py

food-delivery-dataset-proje.ipynb

col2=test\_df.columns.to\_list()

Code

Markdown

Run All

Restart

Clear All Outputs

Variables

Outline

```

1
2 test_df.describe([0.15,0.25,0.35,0.5,0.65,0.75,0.9]).T

```

[ ]

1 df.sort\_values("Delivery\_person\_Ratings",ascending=False).head(20)

[ ]

1 df.sort\_values("Delivery\_person\_Ratings",ascending=True).head(20)

[ ]

1 df.sort\_values("Time\_taken(min)",ascending=False).head(20)

[ ]

1 df.sort\_values("Time\_taken(min)",ascending=True).head(20)

[ ]

1 df.groupby("Weatherconditions")[["Time\_taken(min)", "Delivery\_person\_Age", "Delivery\_person\_Ratings"]].describe().T

2

[ ]

1 df.groupby("Road\_traffic\_density")[["Time\_taken(min)", "Delivery\_person\_Age", "Delivery\_person\_Ratings"]].describe().T

2

[ ]

```

1 df.Type_of_order.value_counts()

```

[85]

```
food-delivery-dataset-proje.ipynb • food_delivery_app.py
food-delivery-dataset-proje.ipynb > col2=test_df.columns.tolist()
+ Code + Markdown | ▶ Run All ⌂ Restart ⌂ Clear All Outputs | 📄 Variables ☰ Outline ...
▶ 1 df.groupby("Type_of_vehicle")[["Time_taken(min)", "Delivery_person_Age", "Delivery_person_Ratings"]].describe().T
2 ⚠
[ ]
```

```
1
2 from datetime import datetime
[88]
```

```
1 df["Order_Date"] = pd.to_datetime(df.Order_Date)
2 ⚠
3 df['year'] = df['Order_Date'].dt.year
4 df['month'] = df['Order_Date'].dt.month
5 df["day"] = df["Order_Date"].dt.day
6
7
[89]
```

```
▶ 1 df ⚠
[ ]
```

```
▶ 1 df["Time_Orderd"] = pd.to_datetime(df.Time_Orderd)
2 ⚠
3 df['time_ord_hour'] = df['Time_Orderd'].dt.hour
4 df['time_ord_minute'] = df['Time_Orderd'].dt.minute
5
[91]
```

```
1 df["Time_Order_picked"] = pd.to_datetime(df.Time_Order_picked)
2 ⚠
```

```
food-delivery-dataset-proje.ipynb • food_delivery_app.py
food-delivery-dataset-proje.ipynb > df
+ Code + Markdown | ▶ Run All ⌂ Restart ⌂ Clear All Outputs | 📄 Variables ☰ Outline ...
```

```
3 df['time_order_picked_hour'] = df['Time_Order_picked'].dt.hour
4 df['time_order_picked_minute'] = df['Time_Order_picked'].dt.minute
5
[92]
```

```
1 df.drop(["Time_Orderd", "Time_Order_picked", "Order_Date"], axis=1, inplace=True) ⚠
[93]
```

```
▶ 1 df.head() ⚠
[ ]
```

```
1 test_df["Order_Date"] = pd.to_datetime(test_df.Order_Date)
2 ⚠
3 test_df['year'] = test_df['Order_Date'].dt.year
4 test_df['month'] = test_df['Order_Date'].dt.month
5 test_df["day"] = test_df["Order_Date"].dt.day
6
7 test_df["Time_Orderd"] = pd.to_datetime(test_df.Time_Orderd)
8
9 test_df['time_ord_hour'] = test_df['Time_Orderd'].dt.hour
10 test_df['time_ord_minute'] = test_df['Time_Orderd'].dt.minute
11 test_df["Time_Order_picked"] = pd.to_datetime(test_df.Time_Order_picked)
12
13 test_df['time_order_picked_hour'] = test_df['Time_Order_picked'].dt.hour
14 test_df['time_order_picked_minute'] = test_df['Time_Order_picked'].dt.minute
15
[95]
```

```
▶ 1 test_df.head() ⚠
[ ]
```

food-delivery-dataset-proje.ipynb • food\_delivery\_app.py

food-delivery-dataset-proje.ipynb > df

+ Code + Markdown | ▶ Run All ↺ Restart ≡ Clear All Outputs | 133 Variables ≡ Outline ...

[98] 1 test\_df.drop(["Time\_Orderd","Time\_Order\_picked","Order\_Date"],axis=1,inplace=True) ⚠

▶ [ ] 1 test\_df.head() ⚠

+ Code + Markdown

[100] 1 df.isna().sum().sum() ⚠

... 0

[101] 1 test\_df.isna().sum().sum() ⚠

... 0

▶ [ ] 1 df.info() ⚠

[ ] 1 num\_list=[i for i in df.select\_dtypes(["int64","float64"])]  
2 cat\_list=[i for i in df.select\_dtypes("object")]

[103] 1 cat\_list ⚠

[ ]

food-delivery-dataset-proje.ipynb • food\_delivery\_app.py

food-delivery-dataset-proje.ipynb > sns.pairplot(df.loc[:,num\_list])

+ Code + Markdown | ▶ Run All ↺ Restart ≡ Clear All Outputs | 133 Variables ≡ Outline ...

▶ [ ] 1 num\_list

[ ]

▶ [ ] 1 k=1  
2 plt.tight\_layout()  
3 plt.figure(figsize=(12,12))  
4  
5 for i in df.loc[:,num\_list]:  
6 plt.subplot(4,4,k)  
7 sns.distplot(df[i])  
8 plt.title(i)  
9 k+=1  
10

[ ]

▶ [ ] 1 df ⚠

[ ]

▶ [ ] 1 for i in cat\_list:  
2 plt.figure(figsize=(13,13))  
3 sns.countplot(x=i,data=df.loc[:,cat\_list])  
4 plt.title(i)  
5  
6  
7

[ ]

▶ [ ] 1 sns.pairplot(df.loc[:,num\_list])

[ ]

```

food-delivery-dataset-proje.ipynb • food_delivery_app.py
food-delivery-dataset-proje.ipynb > sns.pairplot(df.loc[:,num_list])
+ Code + Markdown | ▶ Run All ⏮ Restart ⏮ Clear All Outputs | 📄 Variables 📄 Outline ...
▶ ~
3 df.hist(figsize=(9,9))
4
[ ]

▶ ~
1 sns.lineplot(x=df.month,y=df["Time_taken(min)"],data=df)
[ ]

▶ ~
1 sns.lineplot(x=df.day,y=df["Time_taken(min)"],data=df)
[ ]

▶ ~
1 sns.lineplot(x=df.time_order_picked_hour,y=df["Time_taken(min)"],data=df)
[ ]

▶ ~
1 test_df.select_dtypes("object")
[ ]

▶ ~
1 df.select_dtypes("object")
[ ]

1 for i in cat_list:
2     df[i] = df[i].factorize()[0]
3
[116]

1 for i in cat_list:
2     test_df[i] = test_df[i].factorize()[0]
3

```

```

food-delivery-dataset-proje.ipynb • food_delivery_app.py
food-delivery-dataset-proje.ipynb > sns.pairplot(df.loc[:,num_list])
+ Code + Markdown | ▶ Run All ⏮ Restart ⏮ Clear All Outputs | 📄 Variables 📄 Outline ...
▶ ~
1 df.head()
[ ]

▶ ~
1 plt.figure(figsize=(12,12))
2 sns.heatmap(df.corr(),annot=True,linewidths=0.6,fmt=".2f",cmap="coolwarm");
[ ]

▶ ~
1 df.corr()["Time_taken(min)"].sort_values(ascending=False)
2
3
[ ]

▶ ~
1
2 cor=df.corr()["Time_taken(min)"].sort_values(ascending=False)
3 pd.DataFrame({"column":cor.index,"Correlation with Time_taken(min)":cor.values})
[ ]

▶ ~
1 df
[ ]

1 x=df.drop("Time_taken(min)",axis=1)
2 y=df["Time_taken(min)"]
[123]

1 X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.18,random_state=0)
[124]

```



```

food-delivery-dataset-proje.ipynb • food_delivery_app.py
food-delivery-dataset-proje.ipynb > sns.pairplot(df.loc[:,num_list])
+ Code + Markdown | ▶ Run All ↺ Restart ≡ Clear All Outputs | 📄 Variables ≡ Outline ...

1 X_train.shape
[125]
... (35966, 21)

1 X_test.shape
[126]
... (7896, 21)

1 from catboost import CatBoostRegressor
2 from lightgbm import LGBMRegressor
3 from xgboost import XGBRegressor
[127]

1 ridge=Ridge().fit(X_train,y_train)
2 lasso=Lasso().fit(X_train,y_train)
3 enet=ElasticNet().fit(X_train,y_train)
4 knn=KNeighborsRegressor().fit(X_train,y_train)
5 ada=AdaBoostRegressor().fit(X_train,y_train)
6 svm=SVR().fit(X_train,y_train)
7 dtc=DecisionTreeRegressor().fit(X_train,y_train)
8 (variable) gbm: GradientBoostingRegressor
9
10 gbm=GradientBoostingRegressor().fit(X_train,y_train)
11 lgb=LGBMRegressor().fit(X_train,y_train)
12 catboost=CatBoostRegressor().fit(X_train,y_train)
[ ]

1 models=[ridge,lasso,enet,knn,ada,svm,dtc,rf,xgb,gbm,lgb,catboost]
[130]

```

```

food-delivery-dataset-proje.ipynb • food_delivery_app.py
food-delivery-dataset-proje.ipynb > def ML(y,models):
+ Code + Markdown | ▶ Run All ↺ Restart ≡ Clear All Outputs | 📄 Variables ≡ Outline ...

1 def ML(y,models):
2     y_pred=models.predict(X_test)
3     accuary=r2_score(y_test,y_pred)
4     return accuary
[131]

1 for i in models:
2     print(i,"Algorithm succed rate :",ML("Time_taken(min)",i))
[ ]

1 import optuna
2 from optuna import Trial,visualization,trial
3 from optuna.samplers import TPESampler
4
[7]

1 def return_score(param):
2     model=XGBRegressor(**param).fit(X_train,y_train)
3     y_pred=model.predict(X_test)
4     acc=r2_score(y_test,y_pred)
5     return acc
[8]

1 def objective(trial):
2     param={
3         "n_estimators":trial.suggest_int("n_estimators",10,1000),
4         "max_depth":trial.suggest_int("max_depth",2,15),
5         "learning_rate":trial.suggest_loguniform("learning_rate",0.05,0.7),
6     }
7     return(return_score(param))
[133]

```

food-delivery-dataset-proje.ipynb • food\_delivery\_app.py

food-delivery-dataset-proje.ipynb > def ML(y,models):

+ Code + Markdown ▶ Run All ↺ Restart ≡ Clear All Outputs 📄 Variables ≡ Outline ...

[141] 1 test=test\_df.iloc[0:7896]⚠

[142] 1 test['Time\_taken(min)']:=xgb\_tun.predict(X\_test)  
2 ⚠

▶ ▾ 1 test⚠  
[ ]

food-delivery-dataset-proje.ipynb • food\_delivery\_app.py

food-delivery-dataset-proje.ipynb > def ML(y,models):

+ Code + Markdown ▶ Run All ↺ Restart ≡ Clear All Outputs 📄 Variables ≡ Outline ...

▶ ▾ 1 study=optuna.create\_study(direction="maximize")  
2 study.optimize(objective,n\_trials=100)  
3

[ ]

1 trial=study.best\_trial  
2 print("accuary:{}".format(trial.value))  
3

[ ]

▶ ▾ 1 trial.params⚠  
[ ]

▶ ▾ 1 optuna.visualization.plot\_param\_importances(study)⚠  
[ ]

[138] 1 xgb\_tun=XGBRegressor(n\_estimators=378, max\_depth= 8, learning\_rate= 0.0681643817417908).fit(X\_train,y\_train)⚠

1 y\_pred=xgb\_tun.predict(X\_test)  
2 score(y\_test,y\_pred)

[ ]

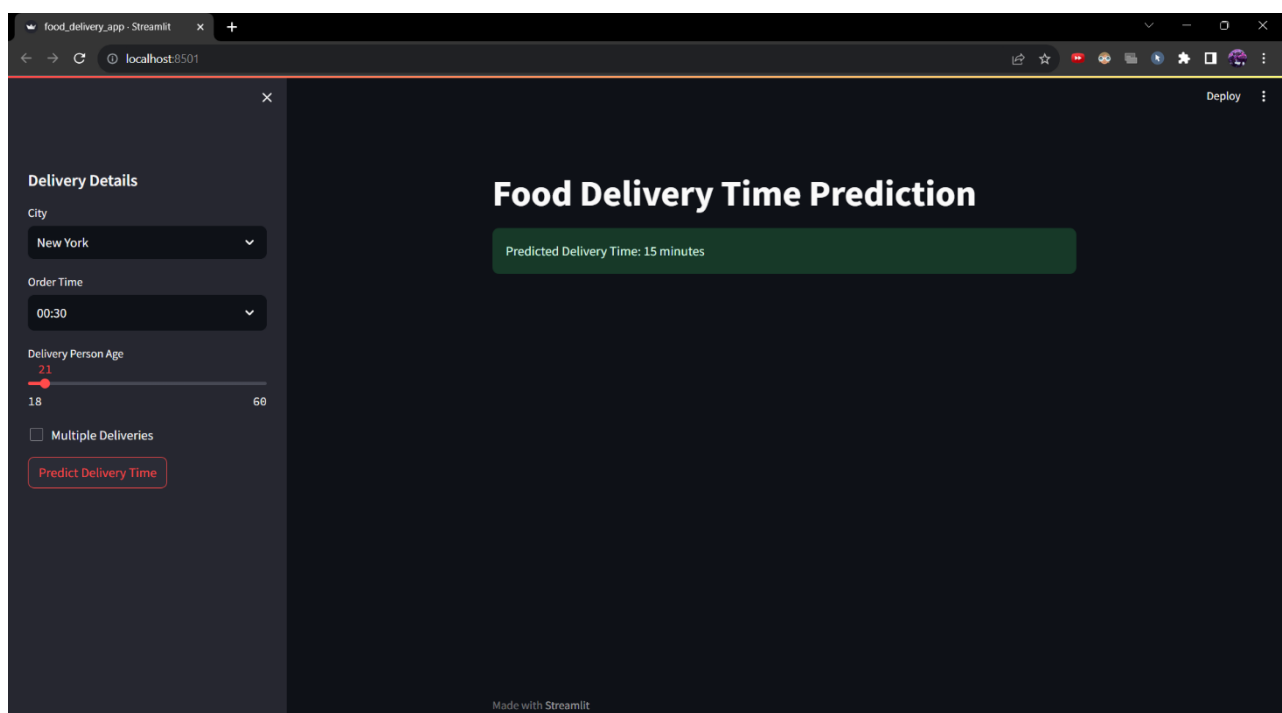
▶ ▾ 1 xgb\_tun.predict(test\_df)[0:100]⚠  
[ ]

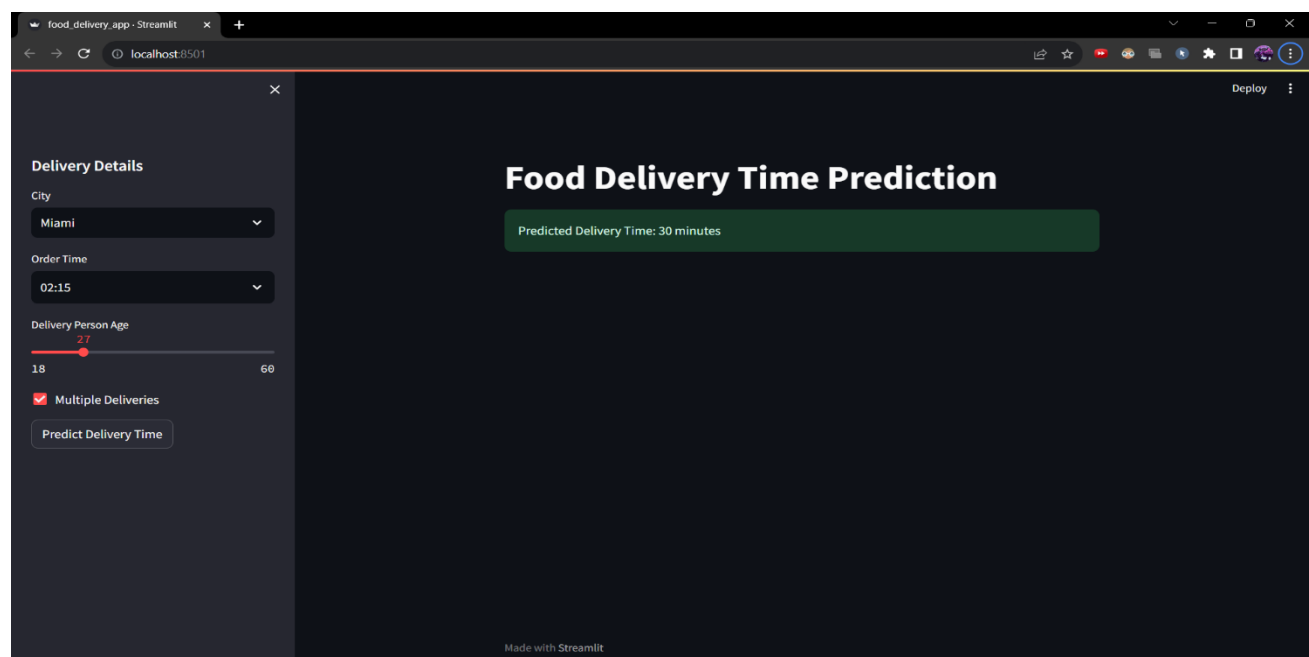
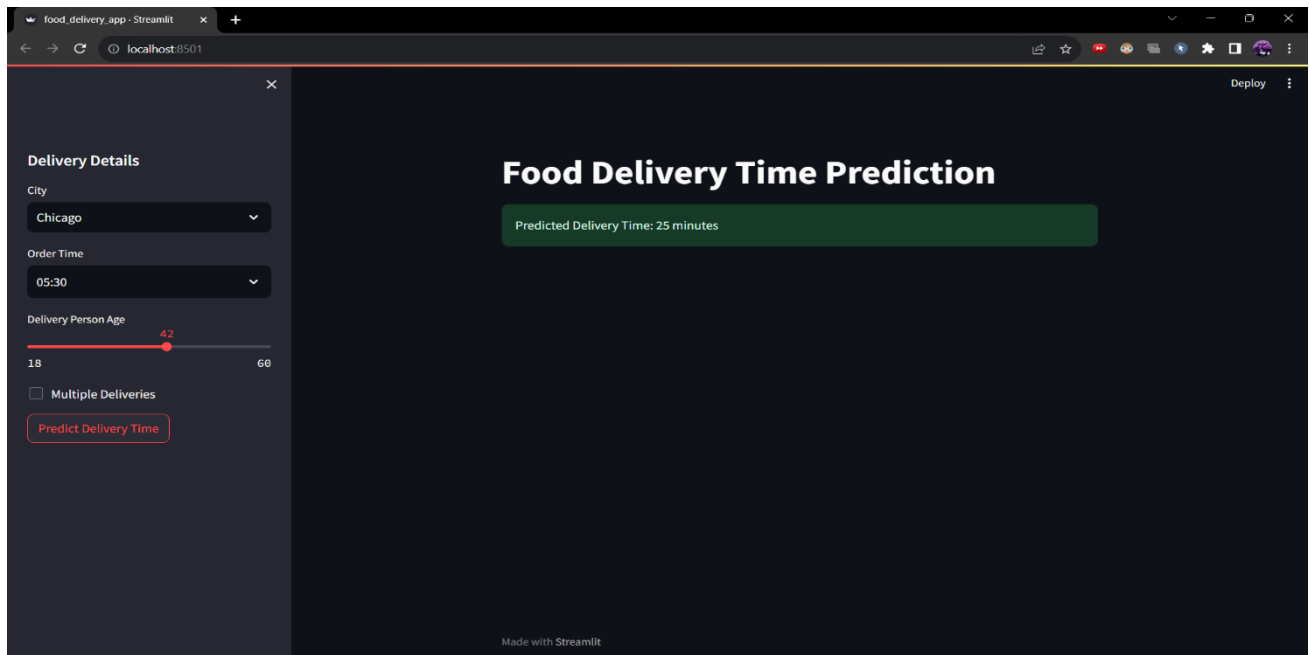
```

food-delivery-dataset-proje.ipynb • food_delivery_app.py •
food_delivery_app.py > main
1  import streamlit as st
2  import pandas as pd
3  from xgboost import XGBRegressor
4  def predict_delivery_time(input_data):
5
6
7      prediction = 25
8
9      return prediction
10
11 def main():
12     st.title("Food Delivery Time Prediction")
13     st.sidebar.header("Delivery Details")
14
15     city_names = ["New York", "Los Angeles", "Chicago", "San Francisco", "Miami", "Seattle"]
16     city = st.sidebar.selectbox("City", city_names)
17
18     order_time = st.sidebar.time_input("Order Time")
19     delivery_person_age = st.sidebar.slider("Delivery Person Age", 18, 60, 30)
20     is_multiple_deliveries = st.sidebar.checkbox("Multiple Deliveries")
21
22     input_data = {
23         "City": city,
24         "Order Time": order_time,
25         "Delivery Person Age": delivery_person_age,
26         "Multiple Deliveries": is_multiple_deliveries,
27     }
28
29
30     if st.sidebar.button("Predict Delivery Time"):
31         predicted_time = predict_delivery_time(input_data)
32         st.success(f"Predicted Delivery Time: {predicted_time} minutes")
33
34 if __name__ == '__main__':
35     main()
36

```

## Code Output





## CONCLUSION

Predicting food delivery times using Machine Learning is a powerful approach to enhance transparency in food delivery services. By leveraging historical data and employing advanced prediction models, we can provide customers with precise delivery time estimates, enhancing their overall experience.

In conclusion, Machine Learning has the potential to revolutionize the food delivery industry, making deliveries more efficient and dependable for customers and service providers alike. The data-driven approach employed in this article highlights the importance of informed decision-making in various domains.

This article has provided a comprehensive guide to building a food delivery time prediction system, and it offers a glimpse into the broader applications of Machine Learning in addressing real-world challenges.

By ensuring that food deliveries are not only prompt but also predictable, we are taking another step toward enhancing the overall customer experience in the food delivery industry.

## REFERENCE

<https://thecleverprogrammer.com/2023/01/02/food-delivery-time-prediction-using-python/>

### Datasets

<https://www.kaggle.com/datasets/gauravmalik26/food-delivery-dataset/data>

<https://www.kaggle.com/datasets/gauravmalik26/food-delivery-dataset?select=train.csv>

<https://www.kaggle.com/datasets/gauravmalik26/food-delivery-dataset?select=test.csv>